Primer Examen Parcial 2022

Parte teórica

1. Se tiene el siguiente código de un programa. Determine si el mismo compilará correctamente, y en caso de ser necesario realice la/s modificación/es necesaria/s para que no se produzcan warning(s) ni errors(s). Indique los mensajes y valores que se imprimirán en la pantalla de salida:

```
#include <iostream>
using namespace std;

class base
{
    public:
        base(void) { num = 1; cout << num << endl; }
        base(int val) { num = val; cout << num << endl; }
        ~base(void) { cout << "Destructor clase base" << endl; }
    protected:
        int num;
};
class miembro
{
    public:
        miembro(void) { cout << "Constructor miembro" << endl; }
        ~miembro(void) { cout << "Destructor miembro" << endl; }
};
class otra_base
{
    public:
        otra_base(void) { num = 10; cout << num << endl; }
        otra_base(int val) { num = val; cout << num << endl; }
        ~otra_base(void) { cout << "Destructor otra_base" << endl; }
    private:
        int num;
        miembro obj;
};
class derivada : private base , public otra_base
{
    public:
        derivada(void) { base::num = 10; otra_base::num = 20; num = 35; }
        derivada(int valor) : base(valor/10) , otra_base(valor*10) { num = 45;}
        ~derivada(void) { cout << "Destructor derivada" << endl; }
        void SetNum(int val) { num = val; }
        int GetNum() { return num; }
    private:
        int num;
};
int main()
{
    derivada obj1,
             obj2 = 20;

    cout << "\nValor de num en obj1 = " << obj1.GetNum() << endl;
    cout << "\nValor de num en obj2 = " << obj2.GetNum() << endl;

    obj1.SetNum(50);
    obj2.SetNum(100);

    cout << "\nValor de num en obj1 = " << obj1.GetNum() << endl;
    cout << "\nValor de num en obj2 = " << obj2.GetNum() << endl << endl;

    return 0;
}
```

2. ¿Qué alcance tiene la declaración de una variable miembro static y por qué?

3. ¿Cuál es el objetivo del modificador volatile?

4. ¿Existe alguna forma de modificar el valor de una variable miembro de un objeto declarado como constante?

5. Indique que se entiende por tipo de dato **reference** y cuáles son sus principales usos. Como se inicializa una referencia a una variable?

6. Indique qué se entiende por **namespaces** (espacios de nombres). ¿Cómo se utiliza?

Parte prácica

Escriba un programa en lenguaje **C++** que permita ejecutar el conjunto de sentencias o tareas mostradas en el siguiente código (almacenado en main.cpp):

```
#include "cRadioReloj.h"

int main()
{
    cReloj clock1,                  // por defecto: 0 0 0
           clock2(3, 4, 50),
           clock3(clock2);

    clock1.setTiempo(23, 58, 59);
    clock1.setMarca("primero");
    clock2.setMarca("segundo");
    clock3.setMarca("reloj copiado");

    cRadio radio1,                  // por defecto: 95.5 FM false
           radio2(103.3),
           radio3(860.0, AM),
           radio4(radio3);

    radio4.setPrendido(true);

    cRadioReloj alarma1(cTime(23, 59, 59)),
                            // reloj=(23,59,59) alarma=(0,0,0) Timbre
                            // alarma_off 100.1 FM radio_off
                alarma2(clock1.getTiempo(), cTime(8, 29, 58));

    alarma1.setPrendido(true);
    alarma1.setAlarma(12, 59, 59);
    alarma2.setBanda(AM);
    // TODO : encender la radio de alarma2

    alarma1.incrementarTiempo();
   cTime tiempo = alarma1.getTiempo();
    cout<<"Hora mostrada en la radio_alarma1: "<<tiempo<<'\n";
    alarma2.incrementarTiempo();
```

```
        tiempo = alarma2.getTiempo();
        cout<<"Hora mostrada en la radio_alarma2: "<<tiempo<<'\n";
        alarma2.incrementarTiempo();
        tiempo = alarma2.getTiempo();
        cout<<"Hora mostrada en la radio_alarma2: "<<tiempo<<'\n";

        if(alarma1.verificarAlarma())
            cout << "La alarma 1 esta prendida" << endl;
        else
            cout << "La alarma 1 esta apagada " << endl;

        return 0;
}
```

Las clases cReloj, cRadio, cRadioReloj y cTime tienen los siguientes atributos privados / protected:

```
class cReloj
{
   private/protected:
     cTime time;
     char *marca;
   public:
     // todos los métodos necesarios
};

enum TipoDeBanda {AM, FM};
class cRadio
{
    private/protected:
        float frecuencia;
        TipoDeBanda banda;
        bool prendido;
    public:
     // todos los métodos necesarios
};

enum TipoAlarma {Musica, Timbre};
class cRadioReloj: public cRadio, public cReloj
{
    Private/protected:
        cTime alarma;
        TipoAlarma tipo;
        bool prendido;
   public:
     // todos los métodos necesarios
};

class cTime
{
    private/protected:
        unsigned int hora;
        unsigned int minuto;
        unsigned int segundo;
    public:
     // todos los métodos necesarios

};
```

Se debe realizar una correcta modularización del código, y el programa debe construirse (build) sin errores (errors) ni advertencias (warnings).

Recuerde gestionar adecuadamente el uso de la memoria dinámica utilizando las funciones proporcionadas por el lenguaje a tal efecto.

Hint: para considerar los métodos que deben implementar las distintas clases considere las sentencias usadas en el main.